Distributed Operating Systems For Combat Systems

S.J. Miller

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Distributed Operating Systems For Combat Systems

S.J. Miller

Information Technology Division
Electronics and Surveillance Research Laboratory

ABSTRACT

Technical Report

This report looks at the current and the future functionality of the distributed operating systems needed to support Naval combat systems.

Approved for public release

DSTO-TR-0060

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Electronics and Surveillance Research Laboratory
PO Box 1500
Salisbury South Australia 5108

Telephone: (08) 259 7053
Fax: (08) 259 5619

AR-008-919
July 1994

APPROVED FOR PUBLIC RELEASE

CONTENTS

LIST OF TABLES

ABBREVIATIONS

AC	Air Controller
APC	Air Picture Compiler
ASW	Anti Submarine Warfare
CO	Command
CPU	Computer Processing Unit
C2	Command and Control
CTG	Commander of Task Group
EER	Execute Engagement Recommendation
ERL	Engagement Recommendation List
ESM	Electronic Support Measures
EW	Electronic Warfare
EWD	EW Director
FCO	Fire Control Operator
FD	Forms Display
FFG	Frigate Guided missile
FGA	Ground Attack Fighter
FPB	Fast Patrol Boat
HMS	Hull Mounted Sonar
IFF	Identification Friend or Foe
IR	Infra Red
LAN	Local Area Network
LRMP	Long Range Maritime Patrol aircraft
MFC	Multi Function Console
MIF	Missile In Flight
NRS	Navigation Radar System
PDMS	Point Defence Missile System
PFE	Post Fire Evaluation
RTF	Ready To Fire
SNDS	Ship Navigation Data System
SPC	Surface Picture Compiler
SSM	Surface to Surface Missile
SWC	Ships Weapon Coordinator
TID	Touch Input Device
TIR	Target Indicator Radar
TWS	Track While Scan
TWT	Travelling Wave Tube
VECTAC	Vector Attack
VME	Virtual Memory Extender
VRTX	Virtual Real Time eXecutive
WAN	Wide Area Network
WC	Weapon Controller
2DR	Two (2) Dimensional Radar

1 INTRODUCTION

Over the last three decades combat systems have developed in unison with the latest computer technology, from a poorly integrated set of sensor and weapon sub-systems, each having some form of computing power requiring substantial operator assistance and over-ride capabilities, to the current highly developed integrated systems incorporating sophisticated command and control functions (C2). Most new combat systems are based on distributed processing architectures and they incorporate redundancy which, together with the improvement in computer technology and associated reliability, allows these combat systems to achieve their requirements for availability, maintainability, reliability and real-time performance(1,2). These system achievements were not possible through the computer hardware and system architecture developments alone. They also relied on advances in computer languages, computer operating systems, and the software development environments.

Information Technology Division's (ITD) Navy sponsored research on Distributed Processing (Task NAV 87/226) concentrated on distributed computing in terms of architectures(1,2), network communications(3), and distributed database issues relating to management(4,5,6,7) and reliability. A test bed system was developed to evaluate database management protocols(8,9) and to validate the simulation results of a number of communication protocols(10,11). In addition, ITD managed a research agreement between the Defence Science and Technology Organisation (DSTO) and the Computing Science Department of the Adelaide University(12), to survey current real-time distributed operating systems, mainly within the University environment, which would be suitable for command and control. Little effort by ITD has so far been directed towards establishing the requirements of a single platform Naval combat system operating system.

This paper looks at the developments in combat system architectures and operating systems, and identifies their operating system requirements, their current limitations, and their applicability to the single platform Naval combat system. Section 2 addresses early combat systems, their centralised computer architecture, and their associated operating systems. Section 3 looks at the trends in combat system development, distributed architectures. and distributed operating system functions. Section 4 addresses the typical functional and performance requirements for a single platform Naval combat system. Section 5 looks at the suitability of current distributed operating systems for use in combat systems. Section 6 looks at the development trends in distributed operating systems and their applicability to the combat system and finally, section 7 provides an overview of distributed operating systems currently being developed.

2 EARLY COMBAT SYSTEMS

2.1 Centralised Architecture

The designers of combat systems that were used in the early 1970s(1,2) attempted to integrate a number of existing sensor, and weapon sub-systems together with a number of operator display, and control, consoles in order to form a C2 system. Each sensor and weapon sub-system, which was basically designed for stand-alone operation, was modified to provide/accept data from a central system and in so doing provide operators with a way of selectively processing data and providing the Commander with a better overall view of the navigation and tactical picture. With this picture, the Commander was able to more effectively provide timely control of the platform's weapons and was able to manoeuvre the platform into the best tactical position. As these early systems used a single centralised computer for the C2 system, a single component failure often meant a temporary loss of the

C2 function, and sometimes a permanent loss of a substantial amount of collected and processed data. It became necessary to build in a casualty mode to allow weapon systems to be operated independently of the centralised computer.

In these early combat systems the central computer was required to process the data in real-time, ie when data became available it was accepted and processed without introducing any significant delay. The computer accepted sensor data, processed it into tracks, and displayed both the processed and the raw data for operator analysis and selection. The computer was notified of new sensor data often by the means of a device interrupt. Target tracks were selected from the processed sensor data, assigned by operators to weapon systems, and weapon/fire control solutions were calculated. Finally, a selected weapon was fired by an operator on the Commander's approval. The processing requirements were controlled by an application program which was loaded into the computer memory at system initialisation. The initialisation, program loading and I/O interface setup, were controlled by the operating system and a real-time program sequencer, known as the executive.

2.2 Real-Time Systems and Programs

A real-time system is one which must meet certain timing restraints (eg process data within a given deadline). These deadlines may be "hard" or "soft". Hard real-time systems are ones that have time critical deadlines where, if the deadline is missed the system integrity or safety is compromised(17). Process control, air traffic control, and combat systems, are typical examples.

In real-time systems, not all computation will be concerned with hard or critical deadlines. Some will have no deadline requirements, while others will merely have soft deadlines to meet(17). A soft deadline is one that if missed, will not compromise system integrity. An example would be the time required for the processing of update data for display purposes. A small delay in this computation would not be noticed by an operator and would have no effect on the system performance.

The combat system with its centralised computer was an early form of a real-time system. A number of sensor and controlled devices were connected to the computer which ran an operating system, and a real-time application program. The application program read and processed input data, and sent the results to the output devices. Sensor data, such as radar and sonar, along with discrete operator entered data, eg "fire push", arrived asynchronously and generated interrupts while, own platform data was obtained through software polling. The responses to these inputs had certain time constraints which were either a "hard" deadline, as in the case of a fire push command (there are minimum target ranges at which weapons can successfully engage), or, just a requirement to meet a "hard" scheduling priority, as in the case of own platform data.

Real-time system programs must exhibit the characteristics of "liveness", "safety", and "timeliness"(18), if they are to meet their relevant deadline constraints. Kalinsky and Ready(18) define "liveness" as demanding a response for every event or stimulus, "safety" as implying the response provided by the software will be that which appears in its specification, and "timeliness"as requiring that the response occurs within a specified rigid time constraint. Whether or not a program meets its time constraints is largely determined by issues such as system interrupt performance, task scheduling policy of the operating system kernel or executive, and inter-task communication.

2.3 Operating Systems

"An operating system is a program that controls the resources of a computer and provides its users with an interface or virtual machine that is more convenient to use than the bare machine"(13). Initially, operating systems did little more than initialise the hardware and prepare the computer for the first job. This process was often referred to as bootstrapping. The operating system kernel, contained a special process to handle the interrupts and a monitor program to support the tasks(14).

The early centralised computer systems incorporated an operating system which supported a multi-level prioritorised interrupt system. Devices requiring immediate attention were given the highest priority interrupt. For example, the early combat systems employed a real-time clock for time stamping data and for controlling task sequencing. The clock interrupt was therefore given the highest priority to ensure it was not missed while higher priority interrupts were being serviced. In these systems a special program known as an "executive, was often used to time sequence tasks by means of counters and flags. The counters and flags were incremented, or set, within a real-time clock interrupt routine. Time constraints were met by designing programs with known and predictable run times, using known hardware response times and code execution times, so that the overall system response met the required time deadlines.

Levi and Agrawala(19) list two properties which they considered distinguish real-time operating systems from traditional operating systems:

a. Bounded time - Execution of programs within a deadline.

b. Design controllability - Should allow a degree of control to the application designer.

These two properties are derived from the need in the real-time environment to execute programs within time deadlines, and show the strong dependency of real-time operating systems on the application requirement. The real-time operating system essentially allocates and manages resources according to applications requirements.

VRTX is an example of a modern real-time operating system kernel. It performs task management, inter-task communication and synchronisation including mutual exclusion, memory allocation, interrupt servicing, and basic clock and character I/O services(18).

3 TRENDS IN COMBAT SYSTEM DEVELOPMENT

3.1 Distributed Architectures

Over the last decade, developments in processor technology, in particular in the semi-conductor field, have brought about vast improvements in the performance, and reductions in size of computers. In addition, there have been similar improvements in the associated memory systems which are now more compact, have an increased capacity of many orders of magnitude, and have significantly lower access times. There have also been developments in computer busses and communication networks, such as the VME bus, and in Local Area Networks (LAN). With these developments, it is now possible to distribute the processing

to the points where it is needed, and provide greater processing capabilities in terms of parallel processing locally, where large quantities of data have to be processed. The general term for this type of processing is "concurrent"(1).

Two basic types of concurrent computer systems have been developed, the parallel processing system and the distributed processor system. Tanebaum(15), suggests that these systems are basically the two extremes of distributed processing.

Distributed systems can be broadly divided into three categories(15):

a. Closely coupled systems,

b. Loosely coupled systems, and

c. Barely coupled systems.

These systems are basically distinguished by the so called "grain of computation" which is defined as the ratio of computation time to communication time(17). If the ratio is below 10, a system is referred to as "closely coupled" (parallel processing). If the ratio is between 10 and 100 a system is termed "loosely coupled" (joined by a LAN) while, a ratio of greater than 100 relates to a "barely coupled" system (one joined by a Wide Area Network (WAN)).

The trend in the combat system arena is towards an integrated, loosely coupled system of sensor sub-systems, a C2 system, and weapon sub-systems. Modern C2 systems employ distributed computing; however, the degree of distribution varies greatly. It can vary from one level of redundancy where a hot standby computer holds a copy of the software, to a fully distributed system where an identical system architecture and software configuration is used in all processing nodes(1). In the latter case, a system node's function is selected by an operator and at initialisation the required task software configuration is identified for scheduling at run-time. Even though the combat system including the C2 functions could be fully distributed, often sensor data processing algorithms need specialised array and parallel processing elements, which are quite different from the general purpose processing required in C2 nodes, in order that the real time data processing requirements can be met. Further, weapon and sensor processing is generally located close to the associated weapon launcher or sensing elements which must be specifically located for best performance.

Distributed processing systems can be designed with characteristics including reliability, availability, fault tolerance, and graceful degradation, making them ideally suited for combat systems. Williams and Catt(16) state that an effective distributed system needs to have the following functions and system attributes:

- Resilience; robust, fault tolerant, and possibly self healing
- Maintainability
- Availability
- Ease of upgrade
- Coordination between distributed application processes

- Efficient use of any parallel processing power resulting in performance improvements
- Management of data flows within the system

Distributed processing systems offer a high degree of flexibility and performance by enabling application tasks to be distributed to available or lightly loaded nodes. In order for a distributed system to operate correctly and reliably, effective communication (eg fast and reliable), and resource coordination and control, must be implemented. Resources including data must be shared across the system. Resource control can be achieved by using a distributed operating system.

3.2 Distributed Real-Time Operating Systems

To the user, a distributed operating system behaves like a centralised operating system but runs on multiple, independent central processing units (CPUs)(13). It must support the distributed processing system characteristics and functions, identified in Section 3.1. In the case of systems running real-time applications, the operating system has the further complication of needing to support the application tasks in a way which ensures their time constraints are met.

Like the centralised real-time operating system, the distributed version must meet the requirements of the application. However, its complexity will be application dependant, and only implements those properties of the distributed architecture necessary to meet its functional requirements. For example, if the application's requirement is simply to distribute processes to the nodes where they are most applicable, and the tasks on different nodes do not need to be synchronised and there are no system requirements for fault tolerance and graceful degradation, then all that is necessary is for a copy of a centralised real-time operating system to be run on each processing node. Any requirement for data update synchronisation and data sharing could be handled by a special data manager(4,5). The manager would take care of the issues of replication, transaction concurrency, and data consistency. Alternatively, when an application requires high reliability, together with fault tolerance and graceful degradation, the operating system may be required to handle distributed task scheduling with the complications of meeting certain time constraints(19). The distributed real-time operating system needs to address processor allocation, task scheduling across the network, synchronisation aspects, task deadlines, and data management. Whether a data management problem or a distributed scheduling problem, the communication network becomes important in the distributed real-time system. The communication system and its inherent reliability (eg error correction process), transmission time, built in time delays, and delivery mechanisms, have considerable affect on a system's performance and whether an application meets its deadlines.

4 A TYPICAL FUNCTIONAL REQUIREMENT FOR A SINGLE PLATFORM NAVAL COMBAT SYSTEM

An Australian Naval ship, for example a frigate, is required to be versatile and capable of convoy support or surveillance prior to a convoy transiting an area. It may also be required to work as part of a task force, or to participate with allied forces(20). To do this it must have a reliable combat system that is suitable for its operational role. In the following paragraphs an example of one

possible operational environment is provided together with a suitable functional configuration for a Naval combat system. Further, an attempt is made to determine the combat system's performance requirements and hence some of the factors which impact on the real-time operating system's requirement.

4.1 Operational Environment

The physical environment is to be restricted to tropical areas adjacent to small and large islands of various topographies, possibly some 200 nm north of Australia, rather than open deep water conditions. For the following scenarios, operational conditions will include moderate weather, and sea state 4, with 15 kts wind and occasional showers. Forces will be required to work 20 nm off shore from islands in areas where a large number of neutral platforms, such as fishing boats of various sizes, in groups or operating singly, civilian aircraft, usually in recognised airlanes, and merchant ships on recognised routes, will be operating. The threat will be from both Fast Patrol Boats (FPB) carrying either 76 mm guns, twin Surface to Surface Missiles (SSM) or both, and Fast Ground Attack aircraft (FGA) armed with rockets, unguided bombs, and a 45 mm cannon. Other Own forces operating in the area are likely to consist of an FFG, an FFG helicopter, a Long Range Maritime Patrol aircraft (LRMP), and possibly a task force.

4.2 Tasks

Ship tasks will be restricted to surveillance and self defence. The ship must contribute to the overall strategic picture and develop a tactical picture within an area of 200 nm by 400 nm. It must locate, shadow, and mark surface forces believed to be operating in the area.

4.3 Typical Engagements

Typically, the ship will be required to enter into self defence engagements with an FGA or an FPB at short notice. There will be support from own forces operating in the immediate area. The FPB may appear from behind a headland and prepare to attack the ship with one or two missiles from a range of 25 km. Similarly, an FGA flying near ground level could pop up from behind a land feature at a radar range of around 35 Kms with the intent of a missile attack.

4.4 Combat System Functional Configuration

One way of meeting the above operational requirement is to provide the ship with an integrated combat system consisting of sensors, weapons and a C2 function. A typical Naval combat system would consist of the following sub-systems:

- Target Indicator Radar (TIR)
- Two Dimensional Radar (2DR)
- Identification Friend or Foe (IFF)
- Hull Mounted Sonar (HMS)
- Electronic Support Measures (ESM)

- Helicopter Transponder System
- Navigation Radar System (NRS)
- Link 11 system
- Gun system
- Point Defence Missile System (PDMS)
- Ship Navigation Data System (SNDS)
- C2 system

The combat system is responsible for three main top level functions:

a. Surveillance,

b. C2, and

c. Engagement.

The data transferred between these top level functions consists of local system track data and Execute Engagement Recommendation (EER) commands.

4.4.1 Surveillance Function

The surveillance function normally operates automatically. However, a provision is provided for manual management intervention. Each sensor produces local track data which is passed to the C2 system automatically. Remote data from Link 11 is placed in a remote track file within the C2 system.

4.4.2 C2 Function

The frigate C2 system is integrated with the sub-systems by means of a combat system data bus, and a video distribution system. A C2 system consists of a number of computer processors and Multi-Function Consoles (MFC) connected with a C2 video distribution and a C2 data bus. It could be a distributed processing system using either a partially replicated or a fully replicated database. However, the architecture chosen must have sufficient redundancy and casualty modes built in to provide reliability and availability even under minor damage conditions.

The C2 system under the control of the commander and MFC operators, is required to perform the following functions:

- Air, surface/sub-surface picture compilation, track compilation and predictions
- Threat evaluation
- Weapon assignment

- Engagement
- Recording and post firing evaluation
- Navigation including setting waypoints, computing Closest Point of Approach and Intercepts
- Aircraft control including VECTAC
- Link 11 processing

The main outputs from the above functions include the system track file, a threat list, and an engagement recommendation queue.

The tactical picture includes local tracks obtained from sensor sub-systems, remote tracks from remote sources via Link 11, and system tracks.

Track compilation correlates local and remote track data at the system level to form common tracks, and selects local tracks which are to be promoted to system tracks. System track data is stored in the system track file.

The Threat Evaluation process includes searching through the system track file and the remote track file for non common tracks (Link 11 data), for threatening tracks, and then selecting a group of, typically, the 20 most dangerous and placing them in a threat priority list. The track priority order listed in the threat list is continually re-evaluated and updated. Quick reaction targets are identified and are given the highest priority.

Engagement Optimisation is an automatic process of evaluating typically the 5 most urgent threats in terms of their engageability, an engagement schedule, and associated engagement times.

4.4.3 Engagement Function

The Engagement function is concerned with four main tasks:

a. Track designation,

b. Target tracking,

c. Ballistic calculation, and

d. Weapon firing

The main outputs of these tasks are:

- Designated targets,
- Target kinematics,
- Aim point, and

- The firing command.

4.5 System Performance

The combat system's performance depends on the aggregate performance of the following sub-systems:

4.5.1 Sensor Performance

4.5.1.1 Navigational Radar

A standard navigation radar for coastal surveillance and surface vehicle detection is provided.

4.5.1.2 2DR

Typically, the 2DR will have a tracking capacity of 100 air, and 50 surface tracks. The aerial rotation rate will be at least 12 RPM, giving a maximum track update rate of 30 tracks per second.

4.5.1.3 TIR

A typical TIR will have a tracking capacity of 20 air, and 10 surface tracks. Its aerial rotation rate will be at least 30 RPM, giving a maximum track update rate of 15 tracks per second.

4.5.1.4 HMS

If a passive sonar system is included, it should be capable of processing at least 10 contacts, surface/sub-surface, providing track information in the form of bearings and frequency data for classification/identification purposes.

4.5.1.5 ESM

Typically, the ESM system will have a processing capability for 15 contacts of either air, surface, or sub-surface origin.

4.5.1.6 Director

A typical director will include sensors such as TV, IR, Laser range finder, and a Travelling Wave Tube (TWT) radar. Both a bearing and range sensor must be activated together. It will also include a video tracker capable of automatic and manual track from either TV or IR input.

The laser range finder will provide a 360 degree all round bearing coverage and an elevation coverage from -25 degrees to 85 degrees.

The director slew rate will be in the order of 35 degrees per second.

The tracking function will track air targets of ranges greater than 1 km through turns of up to 10 G. It will have a primary target tracking capability plus a

secondary target tracking capability when a second target appears in the range gate. The tracking function will be able to accept tracks from the video tracker (IR/TV), laser, and/or radar. It will take no more than 4 seconds after slew and acquisition to lock onto and initiate a target track.

The director passes track kinematic data to the C2 system for display and recording and to the effector, GUN or PDMS, for control.

4.5.2 Weapon Performance

4.5.2.1 GUN

The GUN will have a 5 inch calibre bore and be capable of firing chaff, starshell, fused and non-fused conventional ammunition. It will have a capability of 10 ammunition types and a maximum of 6 magazines. The GUN will be ship attitude stabilised and have a maximum angular velocity rate of 35 degrees/second. The C2 system provides the fire command to the GUN by means of a hard pedal.

4.5.2.2 PDMS

The PDMS will be capable of auto or manual firings of a single missile, or a salvo of 2. It will take no more than 10 seconds to come from the standby to Ready To Fire (RFT) state, and 4 seconds from RTF to Missile In Flight (MIF). The C2 system provides the fire command to the PDMS by means of a soft key.

4.6 Factors Affecting C2 System Performance

A typical modern Naval combat system consists of sensor and weapon sub-systems integrated with a C2 system. The C2 system consists of a number of MFCs and processing units. It may be a distributed system where, the C2 processing is distributed amongst the MFCs, or one where the processing takes place on one or more centralised CPUs. Each of these configurations provide for a different degree of fault and damage tolerance.

The performance of a real-time system is often expressed purely in terms of the system's capacity to process a given amount of input data and to provide the required responses and/or output within critical deadlines. There are three other performance issues that are of high importance in a C2 system. Firstly a C2 system should have in-built functional flexibility. It should be possible for the functional configuration of the C2 system to be selected at startup and changed when online, without loss of data or interrupting current critical processing. Secondly, the system should be fault and damage tolerant. The system should degrade gracefully under fault conditions. To do this, processes must either be dynamically re-distributed in the case of damage or faults, or at least be operator selectable from another operational unit which has an up-to-date and consistent database. Finally, the system should be able to be readily maintained. It should be possible to take faulty elements out of service for repair and reintroduce the repaired elements without loss of data or the loss of a critical process.

The above performance requirements dictate both the hardware and software architectures needed for the C2 system. Both the software and hardware should be modular for ease of maintenance. Each C2 node should have identical hardware and be software configured to provide flexibility and allow dynamic reconfiguration under fault conditions. Data should be duplicated for redundancy. This means a global data manager must be employed. The

database consistency, availability, and management have been addressed previously in the literature(4,5,6,7,8), and the requirements of real-time operating systems and associated process control are addressed in section 5.

4.6.1 C2 Architecture

The C2 architecture adopted should be able to provide the specified performance (Section 4.7). This means some form of system distribution will be required. Many current Naval combat system C2 architectures, only have a partial capability to degrade gracefully in the events of faults or damage. These systems consist of a number of identical operator consoles (MFCs) and one or more C2 processing and database nodes. Each MFC contains local processing for the man machine interface functions including function selection, operation mode selection, and data display. However, the main C2 processing and data management is still often performed centrally with the concession of a second CPU unit being held as a hot spare to take over when required. This configuration does not provide for the gradual degradation required. Instead, in the event of a failed or damaged element, a process may fail and need to be restarted after a benchmark database is loaded onto the stand-by processor. The database retrieval will take valuable time, and may disrupt critical processing such as threat evaluation and the engagement assessment. Further, the retrieved data will be from a previous benchmark and therefore to some extent, stale. These deficiencies can be overcome with a greater degree of distribution (see below).

The required fault tolerance is only possible by distributing the processing to the local nodes where it is required as well as to alternative nodes for redundancy. Further, those portions of the database, if not all of the database, should be replicated at both the nodes performing the processing(1,2) and those nodes holding the backup processing, and a robust data management system(5,6,7) implemented. The requirements for a distributed operating system are not so clear. In the following sections a typical set of operator C2 roles are listed (section 4.6.3), the C2 processing requirements are defined (section 4.6.4), and an example of a combat system operational scenario requiring a quick response is described (section 4.7). These three sections provide a basis for determining the requirements for a real-time distributed operating system.

4.6.2 Multi Function Console

The MFC is the C2 operator interface. All MFCs consist of identical hardware and achieve different functionality through software. An MFC consists of the following input/output devices and display elements:

a. A raster scan monitor for the display of the tactical picture - the Main Tactical Display (MTD),

b. A raster scan monitor for the display of forms and tabular data - the Forms Display (FD),

c. An alphanumeric keyboard with separate numeric keypad for data input,

d. A touch-sensitive input device (TID), with software driven labels - the Multi Function Keyset Panel,

e. Track balls and direct function keys, used for entering positions, track selections for data display etc, and

f. Buzzer alerts.

The above combination of hardware and local software makes the MFC flexible and multi-purpose. The actual MFC functions are implementation dependant. A sub-set of these functions is determined by the operator selected role and are provided through the multi function keyset. This keyset, with its software driven labels, guides an operator through the associated role functions.

4.6.3 Distribution of Tasks

In the typical C2 system, each operator has a set role to perform. However, this role may be extended during fault conditions to include the additional set of activities, previously performed by another operator at a now failed node.

An operator must select the appropriate console "mode" to configure an MFC for a given role. The configuration results in each MFC having the appropriate functions enabled, the multi-function keyset panel suitably labelled, and both the Main Tactical Display and the FD initialised.

In the following example of a Naval single platform C2 system, seven main operator roles and their associated activities are shown (table 1). In this example only a minimal ASW role is anticipated which will be handled by the SPC. If a separate ASW role is required, as would be the case with the addition of a towed array, one or more additional MFCs will be required.

	OPERATOR ROLE	SUMMARY OF ACTIVITIES
CO	Command	Command appreciation of tactical picture and power of veto.
SWC	Ships Weapons Coordinator	Manages Anti-Aircraft Warfare battle, plans and prepares surface engagements and designates targets to FCO.
APC	Air Picture Compiler	Manages air picture for presentation, Correlates link tracks with situation picture, Performs Network Control functions, and acts as Link Controller.
SPC	Surface Picture Compiler	Manages surface picture for presentation, Decision lines eg shipping and air lanes, and collision avoidance.
EWD	EW Director	Analysis of ESM situation picture.
AC	Aircraft Controller	Controls own helicopter and other aircraft and assists APC.
FCO	Fire Control Operator	Controls FCS and assigned Weapon (GUN and PDMS).

Table 1. Operator Roles

Each operator must perform the activities appropriate to the selected C2 functions. The C2 functions appropriate to Naval combat systems include picture compilation, threat evaluation, weapon assignment, engagement, aircraft control, and navigation etc (see section 4.4). These functions are distributed amongst the available MFCs according to the current operational requirement and track load. A typical C2 system could have seven MFCs. An operator will select the MFC role and functions on startup. Many of these functions are common to more than one role (eg both the APC and AC need the functions of sensor selection, air target picture compilation and automatic surveillance). More than one MFC can be allocated to picture compilation (air picture and surface pictures), and weapon control (GUN and PDMS), when required. The MFCs used for Weapon Control (WC), have additional functionality, the firing selection/pedal, and are therefore unique.

FUNCTION	CO	SWC	APC	SPC	EWD	AC	FCO
Threat Evaluation	X	X					
Air Targets picture comp.			X			X	
Surface Targets picture comp.				X			
Surveillance equipment control			X			X	
Management of passive system (bearing lines)			X		X	X	
Time projection	X			X			
Automatic surveillance			X	X		X	
Patterns (sectors, approach patterns, etc)	X	X	X	X	X	X	X
External communication Link 11		X	X	X	X		
Common console functions	X	X	X	X	X	X	X
War diary	X	X	X	X	X	X	X
Sensor selection		X	X	X	X	X	
Surface to surface missile		X					X
Surface to air missile		X					X
Gun fire control		X					X
Chaff planning		X			X		
Operational configuration	X	X					
Technical configuration			X		X		
Tactical recording	X	X	X				
Technical recording			X			X	
Navigation	X	X					
Notice to mariners	X	X		X			
Environ. data management	X					X	
Replay	X	X	X				
Technical supervision			X			X	X

Table 2. Role and Functions

Consequently, two MFC(WC)s are often provided for redundancy, however, both may not be required at the same time. The MFC is configured at mode selection to provide the operator with a user interface which provides a guide through the associated role functions. The relationship between operator roles and functions is shown in table 2 above.

4.6.4 Processing

An operator selects the MFC Mode according to the required role. Each MFC Mode provides facilities for performing the functions applicable to the selected Mode. Many of the functions are common to more than one role, see table 2. Further, some of these functions will use identical processing even though the data inputs may be different (Air and Surface picture compilation).

The results of the processing associated with the operator selected function, whether a display of the results of a calculation or just an acknowledgment of an action, must be provided locally to the operator. However, it is not important where the actual processing takes place, eg locally or on a remote processing node. What is important is that all MFC's must be able to perform in a timely fashion the functions which call the same processing and which may require access to the same or some of the same data during the process. There must be a means of controlling concurrent processing to ensure a process is not interfered with by another process, and that data is not changed while in use by another process.

One way of meeting the C2 system processing requirements is to have the database fully replicated across all processing nodes and an identical software suite, capable of being configured to meet any given role, available at all nodes. There are two main benefits from adopting this software architecture. All processing and data access can take place locally. In addition, similar processes can be taking place concurrently on separate nodes with the only proviso that data flow is strictly controlled to ensure database consistency is maintained across the distributed system. A distributed database manager is required to control the aspects of reliability, concurrency, and global consistency(4).

4.7 Combat System Performance

The overall combat system performance must be assessed against the worst case operational scenario. The actual performance is a combination of the performances obtainable from the Sensor sub-systems, the C2 system and the Weapon sub-systems.

4.7.1 Quick Reaction Scenarios

A possible ship's operational environment was defined in section 4.1. It is within this environment that the ship's combat system must be capable of responding quickly to pop up targets, see section 4.3. Although the above operational environment has been chosen by way of an example, other environments such as those experienced during the GULF war, are also valid. Alternative environments could place a much heavier track processing load on the C2 system, however, the critical quick reaction response required would be similar. Typical operational scenarios requiring quick reaction are described in section 4.7.2.

4.7.1.1 Basic system background functions

a. Establish and maintain air picture:

- detection:- auto in clear areas, manual over land,
- auto correlation (local and remote tracks with system tracks),
- establish civilian air routes,
- system air track loads:

  (1) 2 manual tracks from TIR,

  (2) 8 auto tracks (6 2DR only, 2 held by 2DR and TIR),

  (3) 1 IFF track,

- Remote track loads, 40 tracks at 150 nm by 200 nm area held by Task Force Group units, 20 of which are also commonly held by the 2DR.

b. Establish threat list with pop up targets placed on top of the list.

c. Establish and maintain surface picture:

- auto track 8 targets on TIR with 2 held also on 2DR,
- manual track two small slow moving tracks on TIR,
- 5 neutral remote tracks with updates every 10 minute intervals from LRMP (LRMP surveillance radar track database),
- 10 friendly remote tracks and 5 neutral remote tracks held by the Task Force Group sensors.

d. Establish and maintain Helicopter picture, launch and control helicopter transit.

4.7.1.2 Scenarios

The quick reaction scenarios are based on the typical expected engagements, an FPB and an FGA attack (see section 4.3). Own ship (eg combat system) together with other own forces must be able to recognise and track these threats, and prosecute successful engagements against them, and, at the same time, maintain the tactical picture within the area of interest.

a. Example FPB scenario with own ship in company with an FFG:

- The nearest surface track (12 nm , 270°) increases speed to 30 kts and turns to close on ship, at time 0 sec,

- ESM reports an FPB surveillance radar on the bearing, at time +30 sec
- Request a Track While Scan (TWS) track at +35 sec, established at +45 sec, FFG prepares for a SM1 missile engagement.
- At 9 nm ESM reports an FPB FC radar in sector search and then track at +345 sec, own ship turns away to increase range,
- Director used to acquire FPB and track at +350 sec, and gun assigned to director,
- FPB opens fire on own ship at +410 sec, CTG advised,
- Own ship engages FPB with gun at 9.5 nm at +415 sec and the FFG engages with SM1 at +425 sec,
- FPB turns away at 30 kts, own ship turns towards FPB maintaining fire bursts,
- SM1 misses, and FPB continues to open and weave to avoid gun fire,
- Target continued to be engaged, a new, air threat, detected at +450 sec.

b. FGA and missile attacks:

- 2DR manual track over land bearing 280°, no mode 4 IFF,
- ESM report, search radar racket from aircraft; classification, possible hostile surveillance aircraft at +450 sec,
- Helo reports new small surface track at 30 nm, FFG advises launcher unavailable due to problem, director released from FPB which is tracked manually by TWS to continue engagement at +480 sec,
- 2DR auto track 18 nm on bearing 280°, IFF "no mode 4" and off airlanes at +500 sec,
- ESM reports FGA radar within 5° of one track at +510 sec,
- ESM reports FPB surveillance radar on helo track bearing,
- Two aircraft continue closing at 500 kts ignoring voice warnings,
- Approval to engage two aircraft at +540 sec,
- FCS locks on lead aircraft at 10 nm at +545 sec,
- Two Missiles In Flight (MIF) on lead aircraft at +575 sec,
- Lead aircraft kill, manual acquisition of second aircraft at +590 sec,

- ESM reports SSM missile head in search, second aircraft breaks away at +590 sec,
- Radar detection at 6.5 nm TIR (+590 sec), FFG reports track on this target and assigns gun, Engagement Recommendation List (ERL) accepted at +600 sec,
- Second SSM head in track on separate bearing, FFG reports track,
- MIF on missile 1 at +615 sec,
- Intercept PFE gives kill at +625 sec at range 3 nm,
- FCS track on missile 2 at +630 sec at range 5 nm,
- Missile 2 track in ERL as priority 1, accept ERL at +635 sec,
- MIF on missile 2 at +650 sec,
- Intercept on missile 2 at +660 sec at range 1.5 nm, FPB out of range , cease fire,
- Post attack.

4.7.2 Performance Implications Which Can be Deduced From the Scenarios

The overall C2 performance hinges greatly on the restrictions imposed by human factors such as operator reaction times, Command responses, and the sub-systems, sensor and fire control, performance. It is evident from the air defence scenario (b) above, that the sub-system delays are quite critical for pop up targets such as missiles and aircraft. The major delays are caused by the sensor sub-systems, the FCS, and any operator or command intervention. Additional, smaller delays are introduced by the C2 processing system, however, the critical computational times can be kept to a minimum with the use of task priority scheduling and the inclusion of sufficient processing power at each node.

Pop up targets such as aircraft flying low over headlands, and especially SSMs fired from FPBs, give a combat system little time to react and the delays introduced by the sensor and weapon sub-systems can be quite critical and may result in an unsuccessful engagement. In scenario (b), an FGA is initially detected by ESM over an island headland and manually held by the 2DR at +450 seconds. Another 50 seconds elapses before the two FGAs have cleared the headland and the 2DR has an auto track on one of the aircraft. The 2DR requires a minimum of two clear sweeps (12 RPM sweep rate table 3), to form a track. The second track is identified at +510 seconds. After Command approval to engage the FGA, the target is designated to the FCS at +540 seconds. The FCS requires 5 seconds to lock on and acquire a target. The PDMS is selected to engage the FGA. Each missile fired takes 15 seconds from "Ready to Fire" until MIF. A total delay of 30 seconds is required to bring two missiles to MIF at +575 seconds.

A more critical situation arises in the 2nd portion of scenario (b) when two SSMs are fired from an FPB. The TIR obtains a track on the first SSM (SSM1) at +590 seconds,

range 6.5 nm, and the ERL is accepted at +600 seconds. MIF is achieved 15 seconds later at +615 seconds with an intercept at a range of 3nm at + 625 seconds. A second SSM on a different bearing is also reported before MIF of missile 1. An FCS track for the second SSM (SSM2) is achieved at +630 seconds at a range of 5nm. MIF is achieved by missile 2 on SSM2 at +645 seconds with an intercept at +655 seconds at a range of 1.5nm. There is very little lee-way in the pop up SSM engagements as the sensor sub-system's and weapon sub-system's delays have a considerable impact on whether a successful engagement is achieved.

The major sub-system delays are caused by sensor data gathering systems and weapon system pre-fire and fire sequences. Sensor sub-systems such as the radars, have contact detection and track update rates limited by for example aerial rotation rates (eg 12 rpm for 2DR and 30 RPM for the TIR). Similarly, the director has a maximum slew rate limitation of 5 seconds per 180 degrees, and the PDMS introduces further significant delays in bringing a missile from standby to Ready-To-Fire (RTF), and from RTF to MIF (eg 15 seconds from RTF to MIF). Table 3 lists typical delay times associated with the above sub-systems.

SUB-SYSTEM	FUNCTION	CAUSE	DELAY
2DR	Detection and track update	Aerial rotation rate of 12 RPM (min. of two sweeps)	10 seconds
TIR	Detection and track update	Aerial rotation rate of 30 RPM (min. of two sweeps)	4 seconds
DIR	Lock on	35 degree/second slew rate	5 seconds for 180 degrees
PDMS	Standby to RTF RTF to MIF	Gyro run up Ignition and lift-off	10 seconds 5 seconds

Table 3. Sub-system Delays

The Commander and the operators also introduce significant delays which are often many seconds during a pre-engagement sequence. During the FGA engagement scenario (b), a delay in excess of 10 seconds is introduced by the Commander before issuing his approval to engage both the FGA's. Other operator delays, also of many seconds duration, are introduced during contact classification and identification. A further delay, in the case of the pop-up SSMs of about 10 seconds, is added in accepting the ERL and designating the target to the FCS (eg at +600 seconds for SSM1 and +645 seconds for SSM2).

Bearing in mind all the sub-system, Commander and operator introduced delays, the combat system must reliably process the data in a timely manner. Any loss of data prior to an engagement due to a C2 system element failure, requiring the collection and re-processing of the track and track associated data, will seriously affect the successful prosecution of the engagement. Further, to minimise the FCS target acquisition time,

the target's current co-ordinates and velocity given at designation time, must be accurate to ensure it is acquired quickly.

5 SUITABILITY OF CURRENT DISTRIBUTED OPERATING SYSTEMS TO THE COMBAT SYSTEM

The C2 function within a combat system is required to process quick reaction data, and time critical data, and to calculate control data to meet the real hard deadlines imposed by missile attacks etc. Ship safety/survival issues are of great concern to the Commander. These critical issues depend greatly upon the control provided by the real-time operating system. Currently, a copy of a real-time operating system is distributed to each computing node. Real-time operating system kernels, such as VRTX(18), provide control in terms of task communication and synchronisation by using queues, mail boxes, flags, interrupt services, and clock services. Tasks are given an identification and a priority status to aid the control process. Further, VRTX can be said to be deterministic, as it has a predictable response for all operations which can be determined by simple algebraic formula. However, with all this, modern real-time operating systems such as VRTX basically only provide an interrupt or time sequencing facility and have no means of dynamically selecting task priority or processing according to urgency, required deadline, the deadlines of currently running task processes, or remaining slack time. Slack time can be defined as the difference between a task's required processing time and time to go to its deadline.

The current generation of real-time operating systems and kernels have sufficient functionality for the current generation of combat systems (eg VRTX provides adequate task communication, synchronisation, and has a satisfactory interrupt service response). However in their present form, they provide limited control in terms of deadline selection, real-time response, and distributed task scheduling. There are a number of factors which have a direct bearing on the operating system requirements for the next generation combat system. They are as follows:

a. Degree of reliability and hence the level of graceful degradation required,

b. Requirement for decreased reaction time. and

c. Degree of system autonomy required.

There are certain inter-relationships between the above requirements. For example, to reduce reaction times it is necessary to reduce the amount of operator control. Less operator control gives the system greater autonomy and requires greater reliability. The greater reliability can only be achieved with a greater distribution of data and tasks with the possible addition of dynamic task scheduling and load balancing, eg a truly distributed operating system.

The driving force for the development of improved distributed operating systems is the need to decrease combat system reaction times. There are many external factors which can influence the minimum reaction time of the next generation combat system for RAN platforms. The following factors are most relevant:

- Faster missiles,
- Less detectable targets,

- Cold war (ie increased pressures on military development),
- Superpowers (ie increased pressures on military development due to the existence of opposing powers),
- Conventional warfare/nuclear warfare (ie likely type of warfare),
- RAN's future role, and
- RAN's operational environment

Currently Australia is faced with a low threat situation which, according to the Strategic Outlook, is likely to be maintained. Tensions between superpowers has eased considerably over the last few years resulting in a considerable reduction in world-wide defence spending. With this reduced spending by the superpowers comes a slow down in development, giving some breathing space and lessening the immediate requirement for the next generation combat system.

The operational environment is most likely to affect the RANs combat system requirement. The operational scenario examples provided in section 4.7, show that a frigate operating singly in Australian's northern waters in offshore island situations, may be at risk to a pop up missile attack because of its relatively long combat system reaction time.

To decrease a combat system's reaction time it is necessary to consider the following factors:

- Necessity for a Commander to override,
- Increasing system integration, and
- Improving the functionality of the operating system.

As was shown in the air defence scenario (b), Command and operator decisions provide some of the longest system delays (eg approval for engagement and the acceptance of the ERL). These delays can be reduced by making the combat system more autonomous. But, to achieve greater autonomy there will need to be a better integration between the sensor sub-systems, the C2 function, and the weapon sub-systems. There will need to be considerable improvements in reliable and accurate automatic identification and classification. These functions will need to be more closely integrated with the detection and processing functions, the threat analysis and weapon allocation functions, and the fire control sub-system. To achieve this degree of integration and reliability, the combat systems will need to be more autonomous, have greater distribution of tasks and data, and have an operating system with greater functionality, control, and better real-time response. It may not be possible to totally achieve these requirements without improvements to the detection, tracking, and the identification functions to reduce their associated processing delays.

6 DEVELOPMENT TRENDS IN DISTRIBUTED OPERATING SYSTEMS APPLICABLE TO THE COMBAT SYSTEM

Current real-time operating system research is making distinctions between "hard" and "soft" deadlines(17). It is also concentrating on developing new task scheduling techniques which

incorporate these deadline mechanisms. Chetto(21) addresses both static and dynamic scheduling while Burns(17) looks at:

- the possibility of guaranteeing deadlines for both periodic and non-periodic hard real-time processes on the same processor,
- utilisation of spare time by non-critical processes,
- initial static allocation of processes, and
- dynamic migration of processes in response to changing environment including overload.

Kalinsky and Ready(18) also look at likely future requirements and the possible trends of real-time operating systems and predict that the 21st century kernels will address:

- portability,
- scheduling policies including time deadlines, rate-monotonic, and mixed,
- priority inheritance, and
- load balancing.

All the above functions will become important in future generations of combat systems where, a greater independence from operators will be essential to ensure that critical time constraints are met.

The research of Kalinsky and Ready is being directed towards the distributed, multi-processor environments where the issues of fault tolerance and reliability can be more readily addressed. This research is therefore very relevant to the single platform naval combat system environment.

6.1 Rate-Monotonic Scheduling

Rate monotonic scheduling typically equates priority with urgency. It uses fixed priorities for periodic tasks where, priorities are assigned according to a task's repetition period and not according to its execution time.

6.2 Deadline Scheduling

With deadline scheduling, priorities are assigned dynamically and the highest priority task is executed first. The task priority is gradually raised, using a mathematical formula, as it gets closer to its time deadline. A weighting factor is included in the formula to take into account the task criticality. A very critical task can still be given the highest priority this way.

Deadline scheduling can be used in a fully autonomous combat system for the automatic assignment of weapon systems to targets within their individual weapon engagement window. If a target kill is not achieved by a given weapon system, or the target moves out of the possible engagement window, then an alternative weapon system can be selected. Further, a time deadline (ie time limit of a possible missile firing), could be set to the time difference between the target's time-to-go to a given range and the time required to launch the missile (ie time from RTF to MIF). When the time deadline is exceeded an available weapon system

in the next level of defence can be automatically assigned to take over the engagement. In addition, a task providing target parameters to the weapon system could have its priority raised as the target gets closer to the edge of an engagement window to ensure that the weapon system is provided with updated data prior to "fire push".

6.3 Priority Inheritance

Task priorities become complicated when a higher priority task calls a lower priority task, because blocking situations can result. A means of elevating a task's priority to a new level, as determined by the calling higher priority task, must be implemented. This is called task inheritance. For example, when a high priority task such as the target parameter calculation task, see paragraph 6.2, calls a lower priority task for calculating own platform position. In this case the called task priority would need to be raised to at least that of the calling task if a lock out is to be avoided.

6.4 Load Balancing

Load balancing basically involves distributing tasks across the network nodes to ensure no single node becomes overloaded and that deadlines are guaranteed. This distribution can be achieved by a method termed "bidding". When a task deadline is not able to be met on a given node, other nodes are given the opportunity to bid for the task. A node's "right" to run the task is based on its available processing time.

7 REAL-TIME DISTRIBUTED OPERATING SYSTEMS CURRENTLY BEING DEVELOPED

The distributed real-time operating system field is still an emerging one (i.e. immature). A report by Barter and Maciunas(12) indicates that most of the research is being under taken by Universities and no commercial system is yet available. A considerable portion of the research is directed towards the general distributed environment and not specifically towards real-time environments. A number of the systems proposed have real-time extensions planned(eg Chorus and RT-MACH). Because of the trend towards micro kernels (i.e. operating system kernels with minimal functionality requiring user code to support high level abstractions to support processes, virtual memory, device drivers and inter-process communication), it should be possible to extend many of these system kernels to real-time. Further, much of the current distributed operating system research is being directed towards specific hardware configurations (eg MARS, HARTOS, and HAWK) and is not easily ported to other hardware configurations(12).

An interesting approach has been taken with Dragon Slayer(12,22). It is a system built on top of an existing hardware/software platform and is mainly directed towards replicated fault tolerant file stores. All inter-process communication is done through message passing. HAWK is not truly a distributed system either, as it has been developed by implementing a distributed system on top of a simpler non-distributed real-time operating system. This is similar to the approach taken for the ADDAM distributed database manager(4). This is a proven technique that can be exploited in the short term to achieve distributed control. However, it is probably not a good long term approach due to its reduced real-time performance.

Many of the emerging distributed operating systems are directed towards the Client/Server model. In its simplest form a Client/Server consists of a provider (server) and a collection of users

(clients). The clients request service from the server, which performs the service and possibly returns responses to the client. In a client/server system, communication is based on the International Standards Organisation (ISO) Open System Interface (OSI) reference model. Further, the spare time between a service request and its completion can be used for computation by the client. A system which supports this model is termed asynchronous. The general model can be modified in several ways by varying the number of servers and clients and the binding between them. In the domain of fault tolerant systems the providers of services are replicated and distributed. How relevant these systems are to the single platform Naval combat system will depend on both, the acceptance of new generation high speed LANs which are more suitable for real-time communication, and the development of reliable distributed resource control. Replicated database management mechanisms need to be robust and reliable in the presence of element and network failure and recovery.

8 CONCLUSION

Naval combat systems have grown, due to necessity, in complexity to provide greater reliability, availability, and improved performance to meet the pop up target threat. Along with this increased complexity has come the requirement for distributed computing to provide both increased processing power, and greater reliability. The current generation combat system for the new Australian naval ships, only partly takes advantage of the capabilities of distributed processing. Its C2 function processing is centralised and redundancy is provided by standby processors as back up. These systems are not totally reliable and do not provide fail safe processing in the event of an element failure. Such systems cannot handle failures during C2 critical processing periods, eg pop up air targets, without loss of some processed data. One way of overcoming the above deficiencies is to use a fully distributed C2 system, where C2 functions are distributed across a number of processors and data is also fully replicated across these processors.

Real-time operating systems, like Ready System's VRTX kernel, are suitable for the current generation combat systems that employ a distributed processing real-time environment. Although these operating systems do not provide a deadline capability, distributed data management, or distributed task control, they do, however, provide a deterministic response for all their supported operations. A copy of this type of operating system can be distributed to each computing node for control of a set of functions pre-allocated to the node processors, and a special purpose data manager can be used for data replication control and maintaining database consistency.

Most of the distributed operating system research is currently being under taken by Universities with only a small amount of the research being directed towards real-time operating systems. Some of the general purpose systems have real-time extensions planned (eg CHORUS and RT_MACH). The research is concentrating on distributed task and resource control, load sharing, and some emphasis is being placed on availability and fault tolerance by means of replication. Many real-time systems and extensions are concentrating on scheduling mechanisms which include hard deadline control (eg ART'S development; RT_MACH). The outcomes of the above research are likely to be gradually introduced into new generation commercial real-time operating systems over the next decade.

In the longer term, beyond 2000 AD, as more sophisticated weapons carriers, greater projectile speeds, and better disguises are developed, both sensor system improvements and better system integration will be required to provide systems with more autonomy (less operator intervention). The new generation of distributed real-time operating systems will be required for these systems if they are to have adequate response against an increased threat.

REFERENCES

No.	Author	Title
1	Schapel, J.G.	"A Survey of Distributed Processing Architectures Suitable for Combat Systems". WSRL-TM-8/90, September 1990
2	Miller, S.J.	"Trends in Distributed Processing for Naval Combat Systems". WSRL-TM-22/89, February 1989
3	Schapel, J.G.	"A Review of Data Communications in Combat Systems". ERL-0659-RN, September 1992
4	Miller, S.J.	"Comparison of Real-Time Command and Control Distributed Databases: Geographically Dispersed and Single Platform Naval Systems". WSRL-TM-45/88, October 1988
5	Miller, S.J.	"A Distributed Database Manager Based on ADDAM". WSRL-TM-58/91, October 1991
6	Miller, S.J.	"Dynamic Control of a Replicated and Distributed Database". ITD-92-20, October 1992
7	Miller, S.J.	"A Fully Replicated Distributed Database System". ERL-0719-RN, January 1993
8	Miller, S.J.	"Distributed Processing Test-Bed System: First Interim Report for the DPTBS". WSRL-TM-26-90, September 1990
9	Miller, S.J.	"Distributed Database Management Evaluation Software". ERL-0635-RN, August 1992
10	Scholz, M.L.	"Simulating Local Area Network Protocols with The General Purpose Simulation System (GPSS)". WSRL-TR-45/89, March 1990
11	Scholz, M.L.	"Simulation of the FDDI Network: A Progress Report". Combat Systems Division, Combat Systems Integration, CSI Working Paper 90/01, June 1990

12	Barter, C.J. Maciunas, K.J.	"A Report on Distributed Operating Real-Time Operating Systems for Use in an Integrated Ship Control System", Dept. Computing Science, Adelaide University, May 1992
13	Tanenbaun, A.S. Von Renesse, R.	"Distributed Operating Systems". ACM Computing Surveys, Vol. 17 No 4, Dec 1985
14	Anderson, D.A.	"Operating Systems". Computer, Vol. 14, June 1981
15	Tanebaum, A.S. Von Renesse, R.	"Reliability Issues in Distributed Operating Systems". 6th Symposium on Reliability of Distributed Software and Database Systems, March 1987
16	Williams, D. Catt, R.	"Data Management For Distributed Systems". Presented at Defence Oceanology International '91' by Software Sciences, Ref 8534, March 1991
17	Burns, A.	"Scheduling Hard Real-Time Systems: A Review". Software Engineering Journal, May 1991
18	Kalinsky, D. Ready, J.	"Real-Time Operating System Kernel Technology in the 21st Century". Ready Systems Corporation, USA, 1992
19	Levi, S. Agrawala, A.K.	"On Real-Time Operating Systems". AD-A182-193 (Micro fiche) Defence Technical Information Unit.
20	ANZAC Ship Project NobelTech	"System Software Requirement Specification". 9LV453, June 1992
21	Chetto, H. Chetto, M.	"An Adaptive Scheduling Algorithm for Fault-Tolerant Real-Time Systems". Software Engineering Journal, May 1991
22	Wedde, H.F Korel, B.	"Transparent Distributed Object Management Under Completely Decentralised Control". 9th IEEE International Conference on Distributed Computing, 1989

DISTRIBUTION

	No. of Copies
Defence Science and Technology Organisation	
Chief Defence Scientist)	
Central Office Executive)	1 shared copy
Counsellor, Defence Science, London	Cont Sht
Counsellor, Defence Science, Washington	Cont Sht
Senior Defence Scientific Adviser	1 copy
Scientific Adviser POLCOM	1 copy
Navy Office	
Navy Scientific Adviser	1 copy
Director, Naval Combat Systems Engineering	1 copy
Army Office	
Scientific Adviser, Army	1 copy
Airforce Office	
Air Force Scientific Adviser	1 copy
Defence Intelligence Organisation	
Assistant Secretary Scientific Analysis	1 copy
Mr P. Whitbread, SRS DIO Liaison, Intelligence Systems Group, ITD	1 copy
Department of Defence	
Director General, Communications and Information Systems	1 copy
Aeronautical & Maritime Research Laboratory	
Director	1 copy
Chief Air Operations Division	Cont Sht
Chief Maritime Operations Division	Cont Sht
Electronics & Surveillance Research Laboratory	
Director	1 copy
Chief Information Technology Division	1 copy
Chief Electronic Warfare Division	Cont Sht
Chief Guided Weapons Division	Cont Sht
Chief Communications Division	Cont Sht
Chief Land, Space and Optoelectronics Division	Cont Sht
Chief High Frequency Radar Division	Cont Sht
Chief Microwave Radar Division	Cont Sht
Research Leader Command & Control and Intelligence Systems	1 copy
Research Leader Military Computing Systems	1 copy

DISTRIBUTION

	No. of Copies
Research Leader Command, Control and Communications	1 copy
Manager Human Computer Interaction Laboratory	Cont Sht
Head, Program and Executive Support	Cont Sht
Head Software Engineering Group	Cont Sht
Head, Trusted Computer Systems Group	Cont Sht
Head, Command Support Systems Group	1 copy
Head, Intelligence Systems Group	Cont Sht
Head, Systems Simulation and Assessment Group	Cont Sht
Head, Exercise Analysis Group	Cont Sht
Head, C3I Systems Engineering Group	1 copy
Mr A. Allwright, C3I Systems Engineering Group	1 copy
Head, Computer Systems Architecture Group	Cont Sht
Head, Information Management Group	Cont Sht
Mr D. O'Dea, Information Management Group	1 copy
Head, Information Acquisition & Processing Group	Cont Sht
Mr J. Schapel, Information Acquisition & Processing Group, ITD	1 copy
Author	1 copy
Publications & Publicity Officer ITD	1 copy
Libraries and Information Services	
Australian Government Publishing Service	1 copy
Defence Central Library, Technical Reports Centre	1 copy
Manager, Document Exchange Centre, (for retention)	1 copy
National Technical Information Service, United States	2 copies
Defence Research Information Centre, United Kingdom	2 copies
Director Scientific Information Services, Canada	1 copy
Ministry of Defence, New Zealand	1 copy
National Library of Australia	1 copy
Defence Science and Technology Organisation Salisbury, Research Library	2 copies
Library Defence Signals Directorate Canberra	1 copy
British Library Document Supply Centre	1 copy
Parliamentary Library of South Australia	1 copy
The State Library of South Australia	1 copy
Spares	
Defence Science and Technology Organisation Salisbury, Research Library	6 copies

Department of Defence

DOCUMENT CONTROL DATA SHEET

1. Page Classification	Unclassified
2. Privacy Marking/Caveat (of document)	N/A

3a. AR Number	3b. Laboratory Number	3c. Type of Report	4. Task Number
AR-008-919	DSTO-TR-0060	Technical Report	87/226.3

5. Document Date	6. Cost Code	8. No. of Pages	34
July 1994	803885	9. No. of Refs.	22

10. Title

DISTRIBUTED OPERATING SYSTEMS FOR COMBAT SYSTEMS

7. Security Classification

*	Document	Title	Abstract
	U	U	U

S (Secret) C (Confi) R (Rest) U (Unclass)

* For UNCLASSIFIED docs with a secondary distribution LIMITATION, use (L) in document box.

11. Author(s)

S.J. Miller

12. Downgrading/Delimiting Instructions

N/A

13a. Corporate Author and Address

Electronics & Surveillance Research Laboratory
PO Box 1500, Salisbury SA 5108

13b. Task Sponsor

Navy

14. Officer/Position responsible for

Security:.....N/A.....

Downgrading:.....N/A.....

Approval forRelease:..DESRL.....

15. Secondary Release Statement of this Document

APPROVED FOR PUBLIC RELEASE

16a. Deliberate Announcement

No Limitation

16b. Casual Announcement (for citation in other documents)

[X] No Limitation [ ] Ref. by Author , Doc No. and date only.

17. DEFTEST Descriptors

Distributed Processing
Command and Control Systems
Battlefield Information Systems
Combat Information Centres
Computer Networks

18. DISCAT Subject Codes

19. Abstract

This report looks at the current and the future functionality of the distributed operating systems needed to support Naval combat systems.